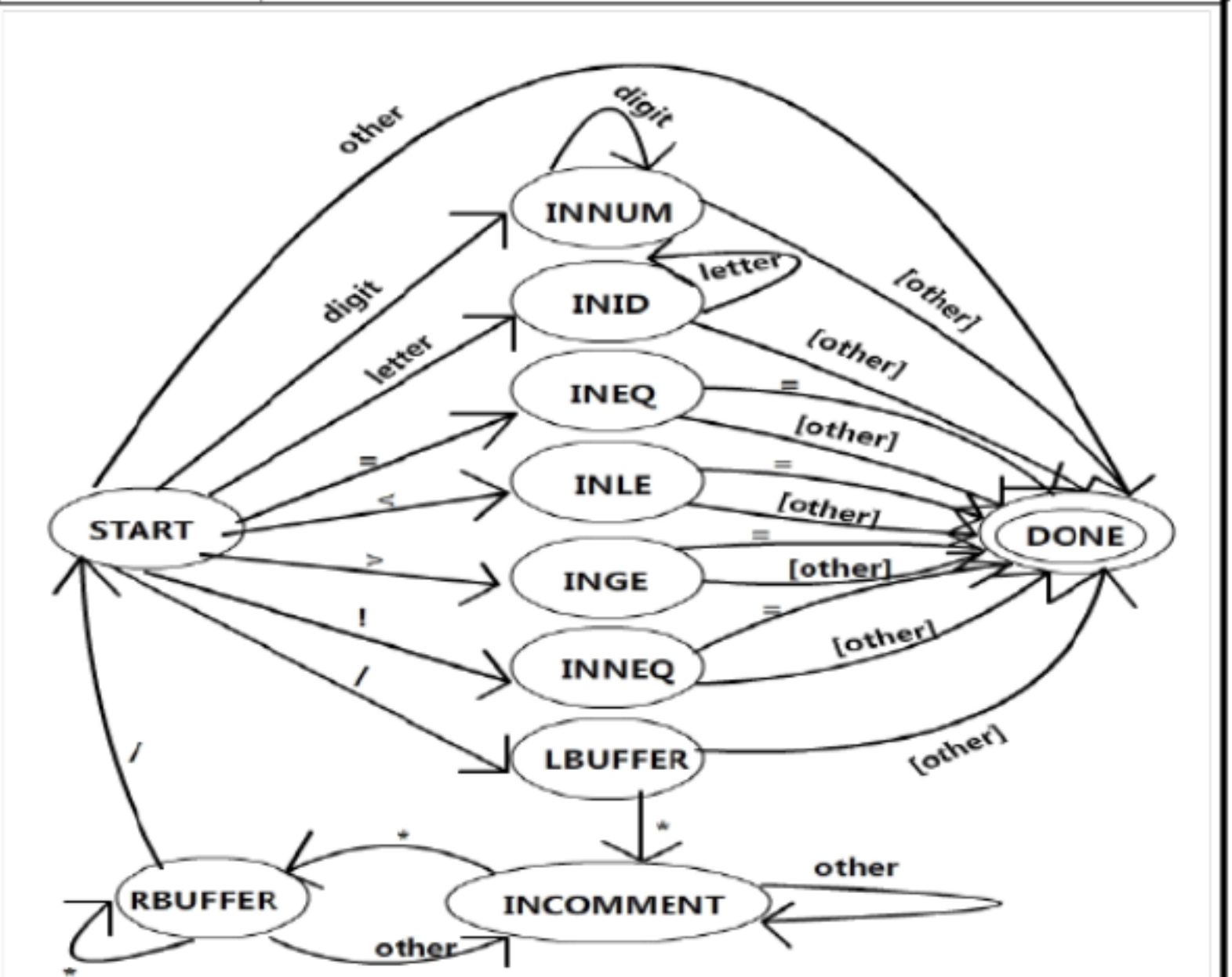
四 川 大 学 计 算 机 学 院、软 件 学 院

实 验 报 告

学号　　姓名：　　专业：

课程名称	编译原理课程设计	实验课时	4
实验项目	手工构造 C-Minus 语言的词法分析器	实验时间	
实验目的、意义	1. 熟悉 C-Minus 语言的词法特点，构造 C-Minus 的 DFA; 2. 设计数据类型、数据结构 3. 通过完成词法分析程序，巩固词法分析知识		
语言特点 正则表达式	注释： /* 注释 */ 关键字： if else int return void while 专用符号： + - * / <<= >>= == != = ; , [ ] ( ) { } ID = letter+ NUM = digit+ letter = [a-z , A-Z] digit = [0-9]		
DFA			

数据类型
数据结构设计
	// 定义数据类型 TokenType
	typedef enum
	{ENDFILE,ERROR,
	IF,ELSE,INT,RETURN,VOID,WHILE,
	ID,NUM,
	ASSIGN,EQ,LT,LE,GT,GE,NEQ,PLUS,MINUS,TIMES,OVER,LPAREN,
	RPAREN,LBRACKET,RBRACKET,LBRACE,RBRACE,COMMA,SEMI
	} TokenType;
	// 定义状态类型
	typedef enum
	{ START,LBUFFER,RBUFFER,INCOMMENT,INNUM,INID,INEQ,INLE,INGE,INNEQ,DONE }StateType;
	// 结构定义
	static struct
	{ char *str;
	TokenType tok;
	} reservedWords[MAXRESERVED]
	={{"if",IF},{"else",ELSE},{"int",INT},{"return",RETURN},{"void",VOID},{"while",WHILE}};

DFA代码映射方法	双层 CASE实现代码映射，外层 CASE关注状态变换，内层 CASE关注输入字符。 外层 CASE一共有 12 个状态： START,LBUFFER,RBUFFER,INCOMMENT,INNUM,INID,INEQ,INLE,INGE,INNEQ,DONE, default ; 内层 CASE判断 getNextChar() 获取的下一个字符使当前状态转换为其他状态。

关键代码分析	TokenType getToken(void)
	{
	int tokenStringIndex=0;

```
TokenType currentToken;

StateType state=START;

int save; //        是否保存到   tokenString

while(state!=DONE)

{

        int c=getNextChar();

        save=TRUE;

        switch(state)

        {



case START:{

                        if(isdigit(c))

                          state=INNUM;

                else if(isalpha(c))

                          state=INID;

                else if((c==' ') || (c=='\t') || (c=='\n'))

                          save=FALSE;

                else if(c=='=')

                          state=INEQ;

                        else if(c=='<')

                                state=INLE;

                else if(c=='>')

                                state=INGE;

                else if(c=='!')
```

```
                state=INNEQ;

            else if(c=='/')

                state=LBUFFER;

            else

            {

                state=DONE;

                switch(c)

                {

                    case EOF:

                        save=FALSE;

                        currentToken=ENDFILE;

                        break;

                    case '+':

                        currentToken=PLUS;

                        break;

                    case '-':

                        currentToken=MINUS;

                        break;

                    case '*':

                        currentToken=TIMES;

                        break;

                    case '(':

                        currentToken=LPAREN;

                        break;
```

```
                case ')':
                    currentToken=RPAREN;
                    break;
                case '[':
                    currentToken=LBRACKET;
                    break;
                case ']':
                    currentToken=RBRACKET;
                    break;
                case '{':
                    currentToken=LBRACE;
                    break;
                case '}':
                    currentToken=RBRACE;
                    break;
                case ';':
                    currentToken=SEMI;
                    break;
                case ',':
                    currentToken=COMMA;
                    break;
                default:
                    currentToken=ERROR;
                    break;
```

```
                    }

                }

                break;

            }

            case LBUFFER:{

                if(c=='*')

                {

                    tokenStringIndex=0;

                    save=FALSE;

                    state=INCOMMENT;

                }


                else if(c==EOF)

                {

                    state=DONE;

                    currentToken=ENDFILE;

                }

                else

                {

                    currentToken=OVER;

                    state=DONE;

                }

                break;

            }
```

```
case INCOMMENT:{
        save=FALSE;
        if(c=='*')
            state=RBUFFER;
        else if(c==EOF)
        {
            state=DONE;
            currentToken=ENDFILE;
        }
        break;
    }
    case RBUFFER:{
        save=FALSE;
        if(c=='/')
            state=START;
        else if(c=='*')
            ;
        else if(c==EOF)
        {
            state=DONE;
            currentToken=ENDFILE;
        }
        else
            state=INCOMMENT;
```

```
                break;



            }

            case INNUM:{

                if(!isdigit(c))

                {

                    ungetNextChar();

                    save=FALSE;

                    state=DONE;

                    currentToken=NUM;

                }

                break;

            }

            case INID:{

                if(!isalpha(c))

                {

                    ungetNextChar();

                    save =FALSE;

                    state=DONE;

                        currentToken=ID;

                }

                break;

            }

            case INEQ:{
```

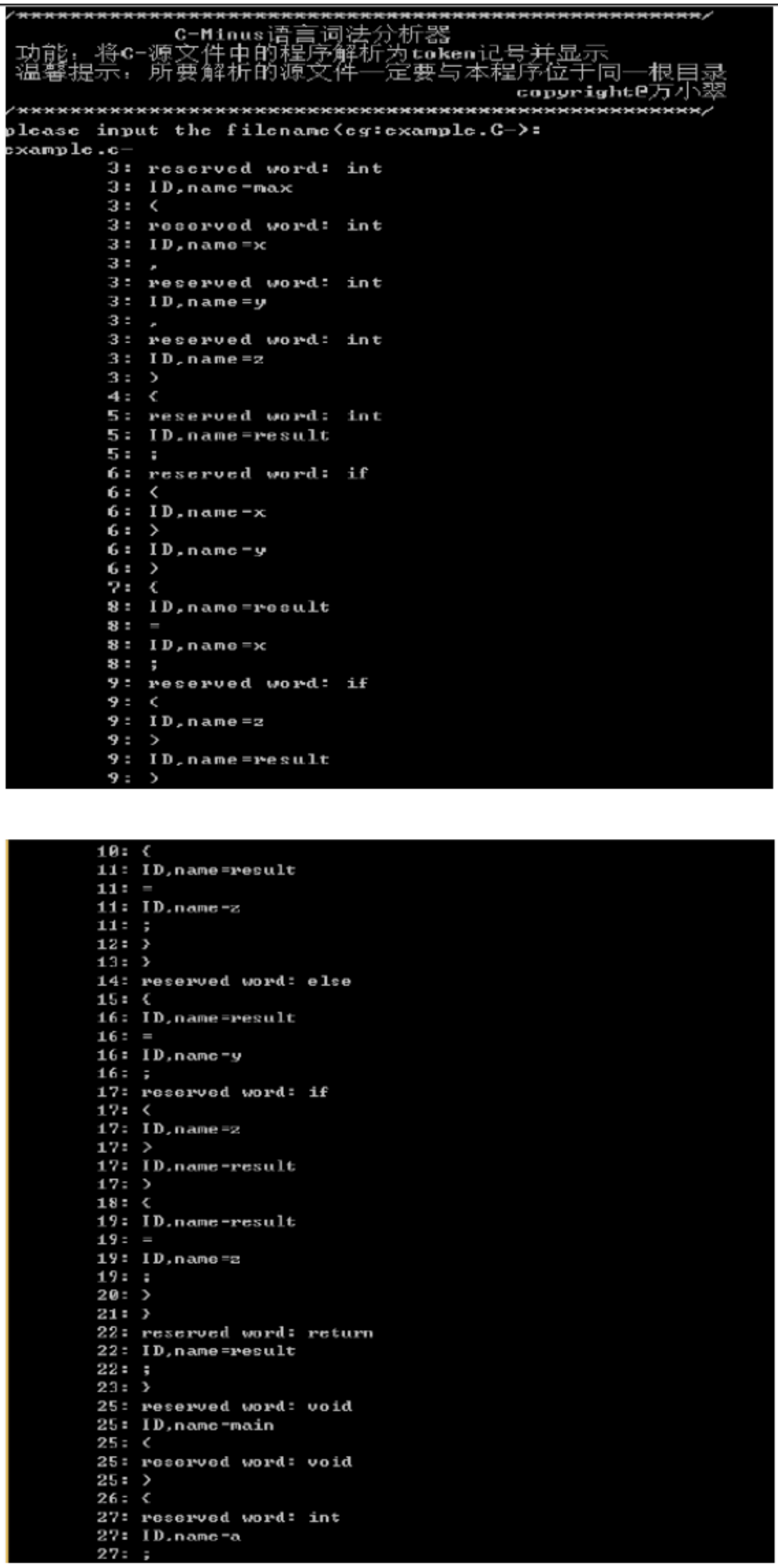
```
                    if(c=='=')
                    {
                        state=DONE;
                        currentToken=EQ;
                    }
                    else
                    {
                        ungetNextChar();
                        save =FALSE;
                        state=DONE;
                        currentToken=ASSIGN;
                    }
                    break;
                }
                case INLE:{
                    if(c=='=')
                    {
                        state=DONE;
                        currentToken=LE;
                    }
                    else
                    {
                        ungetNextChar();
                        save =FALSE;
```

```
                    state=DONE;

                    currentToken=LT;

                }

                break;

            }

            case INGE:{

                if(c=='=')

                {

                    state=DONE;

                    currentToken=GE;

                }

                else

                {

                    ungetNextChar();

                    save =FALSE;

                    state=DONE;

                    currentToken=GT;

                }

                break;

            }

            case INNEQ:{

                if(c=='=')

                {

                    state=DONE;
```

```
                currentToken=NEQ;

            }

            else

            {

                ungetNextChar();

                save =FALSE;

                state=DONE;

                currentToken=ERROR;

            }

            break;

        }

        case DONE:{



            break;

        }

        default:{

            fprintf(listing,"Scanner Bug:state=%d\n",state);

            state=DONE;

            currentToken=ERROR;

                break;

            }

        }



    if((save) && (tokenStringIndex<=MAXTOKENLEN))
```

```
            tokenString[tokenStringIndex++]=(char)c;



        if(state==DONE)

        {

           tokenString[tokenStringIndex]='\0';

              if(currentToken==ID)

                currentToken=reservedLookup(tokenString);

                 }

}



       if(TraceScan){

              fprintf(listing,"\t%d: ",lineno);

                 printToken(currentToken,tokenString);

       }



       return currentToken;

}
```

实验结果截图


C-Minus语言词法分析器
功能：将C-源文件中的程序解析为token记号并显示
温馨提示：所要解析的源文件一定要与本程序位于同一根目录
copyright@万小翠
please input the filename(eg:example.C-):
example.c-
3: reserved word: int
3: ID,name=max
3: (
3: reserved word: int
3: ID,name=x
3: ,
3: reserved word: int
3: ID,name=y
3: ,
3: reserved word: int
3: ID,name=z
3:)
4: {
5: reserved word: int
5: ID,name=result
5: ;
6: reserved word: if
6: (
6: ID,name=x
6:)
6: ID,name=y
6:)
7: {
8: ID,name=result
8: =
8: ID,name=x
8: ;
9: reserved word: if
9: (
9: ID,name=z
9:)
9: ID,name=result
9:)
10: {
11: ID,name=result
11: =
11: ID,name=z
11: ;
12: }
13: }
14: reserved word: else
15: {
16: ID,name=result
16: =
16: ID,name=y
16: ;
17: reserved word: if
17: (
17: ID,name=z
17:)
17: ID,name=result
17:)
18: {
19: ID,name=result
19: =
19: ID,name=z
19: ;
20: }
21: }
22: reserved word: return
22: ID,name=result
22: ;
23: }
25: reserved word: void
25: ID,name=main
25: (
25: reserved word: void
25:)
26: {
27: reserved word: int
27: ID,name=a
27: ;

```
        28: reserved word: int
        28: ID,name=b
        28: ;
        29: reserved word: int
        29: ID,name=c
        29: ;
        31: ID,name=a
        31: =
        31: ID,name=input
        31: (
        31: )
        31: ;
        32: ID,name=b
        32: =
        32: ID,name=input
        32: (
        32: )
        32: ;
        33: ID,name=c
        33: =
        33: ID,name=input
        33: (
        33: )
        33: ;
        34: ID,name=output
        34: (
        34: ID,name=max
        34: (
        34: ID,name=a
        34: ,
        34: ID,name=b
        34: ,
        34: ID,name=c
        34: )
        34: )
        34: ;
        40: }
        41: EOF
Press any key to continue
```

词法分析程序的输出和输入：词法分析程序的功能是读入源程序，输出单词符号。单词符号是程序设计语言的本语法符号，程序设计语言的单词符号一般分为如下几种：关键字，标示符，常数，运算符，界符，单词的输出是二元式的形式，需要知道二元式的表示方法，把得到的二元式写入输出文件。

总结

实验注意事项：

1. 试验中在设计注释部分的解析时，因为 C-Minus 的注释符是四个字符组成，设计 DFA时设计了两个中间态， 用来判断状态转换； 在代码中， 如果由中间态转换为 INCOMMEN状态，注意字符回退和 save 置 false
2. 在判断运算符 <，<=，>，>=，！=时，第二字符是’ =’可成功识别出运算符，第二字符是其他字符时也可能是合法符号，注意字符回退与 token 判断。

参考资料：《编译原理及实践 / 编译器设计方案》

指导老师 评　　议	成绩评定：　　指导教师签名：